ISSN 2278 – 0211 (Online)

RBAM with Constraint Satisfaction Problem in Role Mining

· **B. Shuriya**
Assistant Professor, RVS Technical Campus- Coimbatore, Coimbatore, India
S. Thenmozhi
M. E. Graduate, Coimbatore, India

Abstract:
The main areas of research related to access control concern the identification of methodologies and models. With the ever-increasing number of users and IT systems, organizations have to manage large numbers users permissions in an efficient manner. Role-based access control is the most wide-spread access control model. Yet, companies still find it difficult to adopt RBAC because of the complexity of identifying a suitable set of roles. Roles must accurately reflect functions and responsibilities of users in the organization. When hundreds or thousands of users have individual access permissions, adopting the best approach to engineer roles saves time and money, and protects data and systems. Among all role engineering approaches, searching legacy access control systems to find de facto roles embedded in existing permissions is attracting an increasing interest. Data mining techniques can be used to automatically propose candidate roles, leading to a class of tools and methodologies referred to as role mining. The user role assignment is framed using RBAM algorithm with CSP Technique.

Keywords: *RBAM, CSP, Role mining*

1. Introduction

In this context, role-based access control (RBAC) [2] has become the norm for managing entitlements within commercial applications. RBAC simplifies entitlement management by using roles. A role uniquely identifies a set of permissions, and users are assigned to appropriate roles based on their responsibilities and qualifications. When users change their job function, they are assigned new roles and old roles are removed from their profile. This results in users' entitlements matching their actual job functions. While RBAC is not a panacea for all ills related to access control, it offers great benefits to users managers and administrators, especially non-technical people. First, RBAC helps business users define security policies [5].
Second, RBAC implements the security engineering principles that support risk reduction, such as separation of duties (SoD) and least privilege [3]. Finally, roles minimize system administration effort by reducing the number of relationships among users and permissions [1]. Despite the widespread adoption of RBAC-oriented systems, organizations frequently implement them without due consideration of roles. To minimize deployment effort or to avoid project scope creep, organizations often neglect role definition in the initial part of the deployment project. Very often, organizations do not invest enough time to define roles in detail; rather, they define high-level roles that do not reflect actual business requirements. The result of this careless role definition process is that deployed RBAC systems do not deliver the expected benefits. Additionally, it also leads to role misuse [3].
This is the main reason why many organizations are still reluctant to adopt RBAC. The role engineering discipline [4] addresses these problems. Its aim is to properly customize RBAC systems in order to capture the needs and functions of the organizations. Yet, choosing the best way to design a proper set of roles is still an open problem. Various approaches to role engineering have been proposed, which are usually classified as: top-down and bottom-up. Top-down requires a deep analysis of business processes to identify which access permissions are necessary to carry out specific tasks.
Bottom-up seeks to identify de facto roles embedded in existing access control information. Since bottom-up approaches usually resort to data mining techniques, the term role mining is often used. In practice, top-down approaches may produce results that conflict with existing permissions, while bottom-up approaches may not consider the high-level business structure of an organization [6]. For maximum benefit, therefore, a hybrid of top-down and bottom-up is often the most valid approach.

1.1. User- Role Assignment

UP $\subseteq$ USERS × PERMS, the set of the existing user-permission assignments to be analyzed;
perms: USERS $\rightarrow$ 2 PERMS , the function that identifies permissions assigned to a user. Given u $\in$ USERS, it is defined as perms(u) = {p $\in$PERMS | $\langle u, p\rangle \in$ UP}.

users: PERMS → 2 USERS , the function that identifies users that have been granted a given permission. Given p ∈ PERMS, it is defined as users(p) = {u ∈ USERS | ⟨u, p⟩ ∈ UP}.
System Configuration-ϕ = ⟨USERS, PERMS, UP⟩
RBAC System-ψ = ⟨ROLES, UA, PA, RH⟩
Lemma :Given r1 ,r2 ∈ ROLES such that r2 r1 , the confidence between r1 ,r2 is given by the ratio between supports of child and parent roles: confidence(r2 r1) = support(r2)/support(r1). PROOF By definition, confidence(r2 r1) is equal to: |auth_users(r2)| |auth_users(r1)| · |USERS| |USERS| = support(r2) support(r1) for any given role pair r1 ,r2 .
The administration cost of the role-set built upon the PERMS lattice is neither a maximum nor a minimum of the cost function. In fact, it is possible to increase the cost by increasing the number of role-user relationships. For example, let PERMS = {1, 2, 3} so that ROLES = { {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3} }. If the role {1, 2, 3} is removed from ROLES, a combination of the remaining candidate roles must be used to cover its permissions, such as {1, 2} and {1, 3}. This doubles the number of relationships in UA. Depending on α, β, γ, δ, c(r) and the number of users assigned to {1, 2, 3}, this could increase the cost even if ROLES and PA are smaller. Moreover, the cost is greater than the optimal. In fact, if we delete all roles representing combinations of permissions not possessed by any user, the cardinality of ROLES and PA diminishes while UA remains the same. If c(r) ≥ 0, the cost diminishes as well.

- Pattern Identification in Users' Entitlements:
    - Enumerating Candidate Roles
    - Minimizing the Effort of Administering RBAC
- Devising Meaningful Roles:
    - Measuring the Meaning of Roles
    - Visual Role Mining
- Taming Role Mining Complexity:
    - Splitting Up the Mining Task
    - Stable Roles
    - Imputing Missing Grants
- The Risk of Unmanageable Roles:
    - The Risk of Meaningless Roles
    - Ranking Users and Permissions

2. RBAM

RBAM-purge procedure

```
1: procedure RBAM-purge (Rk−1 , Hk , Hk−1 ,PA,UA, ¯σ, ¯τ, ¯υ)
2: {Remove from parents the users also assigned to children}
3: UA ← {⟨u,r⟩ ∈ UA | u 6∈ S h∈Hk :h.prnt=r ass_users(h.child)}
4: for all r ∈ Rk−1 do
5: r.act_supp ← |{⟨u,r ′ ⟩ ∈ UA | r ′= r}|/|USERS|
6: end for
7: {Identify removable roles with low support}
8: Δ ←  r ∈ Rk−1 | r.act_supp = 0 ∨ r.act_supp ≤ σ¯ (k − 1) + τ¯ + υ¯c(r) ∧
9: r.supp · |USERS| =   S h∈Hk−1 :h.child=r ass_users(h.prnt)
10: {Remove roles with low support}
11: for all r ∈ Δ do
12: {Transfer only direct hierarchies}
13: for all hp ∈Hk−1 , hc ∈Hk : hp .child=hc .prnt=r do
14: if ∄h ′∈ Hk : h ′ .child = hc .child ∧ h ′ .prnt 6∈ Δ ∧
15: ∧ ass_perms(h ′ .prnt)⊇ass_perms(hp .prnt) then
16: h.prnt ← hp .prnt
17: h.child ← hc .child
18: h.conf ← hp .conf · hc .conf
19: Hk ← Hk ∪ {h}
20: end if  21: end for
22: {Transfer users to parents, then remove r}
23: UA ← {⟨u,r ′ ⟩ | ∃h ∈ RH,u ∈ USERS : h.prnt = r ′ ∧ h.child = r ∧ ⟨u,r⟩ ∈
UA}
24: for all r ′∈ {h.prnt | h ∈ RH ∧ h.child = r} do
25: r ′ .act_supp ← |{⟨u,r ′′⟩ ∈ UA | r ′′= r ′ }|/|USERS|
26: end for
27: Rk−1 ← Rk−1 \ {r} 28: Hk−1 ← {h ∈ Hk−1 | h.child 6= r}
29: Hk ← {h ∈ Hk | h.prnt 6= r}
30: PA ← {⟨p,r ′ ⟩ ∈ PA | r ′ 6= r}
31: UA ← {⟨u,r ′ ⟩ ∈ UA | r ′ 6= r}
32: end for 33: return ⟨Rk, Rk−1 , Hk , Hk−1 ,PA,UA⟩
35: end procedure
```

Figure 1: RBAM-purge procedure

It represents the union of all the sets Rk . For each r ∈ ROLES are identified: • r.supp: role r support; • r.act_supp: role r actual support; • r.degree: the number of permissions assigned to r. ▲ The set RH that hierarchically links candidate roles to one another. It represents the union of all sets Hk . This means that only direct relationships are determined. For each h ∈ RH are identified: • h.prnt and h.child: parent and child roles hierarchically related; • h.conf: confidence value between roles. ▲ The set PA. This set merely correlates candidate roles with their assigned permissions. ▲ The set UA. It contains the proposed role-user assignments. At the end of step k, relationships between users and permissions assigned to the level-k roles are added to the set.

3. CSP Technique

A Constraint Satisfaction Problem (CSP) consists of the following: • a set of n variables V = {x1. . . xn}. • discrete, finite domains for each of the variables D = {D1, . . . , Dn}. • a set of constraints R = {R1, . . . , Rm} where each Ri(di1, . . . , dij) is a predicate on the Cartesian product Di1 × · · · × Dij that returns true iff the value assignments of the variables satisfies the constraint. The problem is to find an assignment A = {d1, . . . , dn|di ∈ Di} such that each of the constraints in R is satisfied.

```
procedure initialize di ← random d  Di ;
pi ← sizeof(neighbors) + 1;
mi ← true;
mediate ← false;
add xi to the good list;
send (init, (xi , pi , di , mi , Di , Ci)) to neighbors;
initList ← neighbors;
end initialize;
when received (init, (xj , pj , dj , mj , Dj , Cj)) do Add (xj , pj , dj , mj , Dj , Cj ) to agent view;
if xj is a neighbor of some xk ∈ good list do add xj to the good list;
add all xl ∈ agent view ∧ xl ∈/ good list that can now be connected to the good list;
pi ← sizeof(good list);
end if; if xj ∈/ initList do send (init, (xi , pi , di , mi , Di , Ci)) to xj ;
else remove xj from initList;
check agent view; end do;
```

Figure 2: APO procedures

CSP has been shown to be NP-complete, making some form of search a necessity. Asynchronous Partial Overlay As a cooperative mediation based protocol, the key ideas behind the creation of the APO algorithm are

- Using mediation, agents can solve subproblems of the DCSP using internal search.
- These local sub problems can and should overlap to allow for more rapid convergence of the problem solving.
- Agents should, over time, increase the size of the subproblem they work on along critical paths within the CSP. This increases the overlap with other agents and ensures the completeness of the search.

4. Conclusion

A wide range of users, including IT administrators, business-line managers, and human resources, should feed this process. Most important, the alignment between business and IT is of utmost importance. Second, we demonstrated that the workload of security analysts and role engineers can largely be alleviated via automated approaches to role engineering. The redundancy is removed within user-permission assignments, thus leading to improved mining algorithm performances. Then estimated the minimum number of roles identifiable in the given dataset, hence allowing for the implementation of fast, approximating role mining algorithms.

5. References

1. Colantonio, R. Di Pietro, A. Ocello, and N. V. Verde. A formal framework to elicit roles with business meaning in RBAC systems. In Proceedings of the 14th ACM Symposium on Access Control Models and Technologies, SACMAT ’09, pages 85–94, 2009.
2. American National Standards Institute (ANSI) and InterNational Committee for Information Technology Standards (INCITS). ANSI/INCITS 359-2004, Information Technology – Role Based Access Control, 2004.
3. E. Celikel, M. Kantarcioglu, B. Thuraisingham, and E. Bertino. A risk management approach to RBAC. Risk and Decision Analysis, 1(2):21–33, 2009. IOS Press.
4. E. J. Coyne. Role-engineering. In Proceedings of the 1st ACM Workshop on Role-Based Access Control, RBAC ’95, pages 15–16, 1995.
5. V. Gligor. RBAC security policy model, preliminary draft report. Technical report, R23 Research and Development Department of the National Security Agency, 1995.
6. A. Kern, M. Kuhlmann, A. Schaad, and J. Moffett. Observations on the role life-cycle in the context of enterprise security management. In Proceedings of the 7th ACM Symposium on Access Control Models and Technologies, SACMAT ’02, pages 43–51, 2002.